Dominik Wawszczak
Student ID Number: 440014
Group Number: 1

Parameterized Algorithms
Assignment 2

Problem 1

If G is not connected, we can analyze each connected component independently, subtracting the combined sizes of all other components from k. Thus, for the remainder of the solution, we assume that G is connected.

If there exists a subset of vertices X such that $|X| \leqslant k$ and $G \setminus X$ is a tree, then $\mathrm{tw}(G) \leqslant k+1$, as $\mathrm{tw}(G \setminus X) = 1$, where tw denotes treewidth. We will use the algorithm presented in the lecture, which runs in time $27^l \cdot l^{\mathcal{O}(1)} \cdot n^2$, where l is the target treewidth and n is the number of vertices. We apply it to G with $l = k+1$. The algorithm will yield one of two possible outcomes:

1. Confirmation that $\mathrm{tw}(G) > k+1$.

   In this case, we conclude that no such subset X exists.

2. A tree decomposition of width at most $4k+8$.

   The remainder of the solution focuses on this scenario.

We now proceed with dynamic programming, assuming the decomposition is nice. For any subtree H of the decomposition and its boundary ∂H, let $f : V(\partial H) \to \{0, \ldots, 4k+9\}$. Define $\mathrm{dp}_H(f)$ as the size of the minimum set $Y \subseteq V(H \setminus \partial H)$ satisfying the following conditions:

1. $H \setminus \big(Y \cup f^{-1}(0)\big)$ is a forest,
2. for all $u, v \in V(\partial H) \setminus f^{-1}(0)$, u and v are in the same connected component of $H \setminus \big(Y \cup f^{-1}(0)\big)$ if and only if $f(u) = f(v)$.

If no such set Y exists for a given f, we define $\mathrm{dp}_H(f) = \infty$.

To compute dp_H, we consider the following cases:

1. $H = \mathrm{introduceVertex}(H', u)$

   Here, $\mathrm{dp}_H(f) = \min(\{\mathrm{dp}_{H'}(f') \;:\; f = f'[u \mapsto f(u)] \;\wedge\; f'^{-1}(f(u)) = \emptyset\})$, where $\min(\emptyset) = \infty$. We use the notation $f[a \mapsto b]$ to denote the function g defined by:

$$g(x) \;=\; \begin{cases} b, & \text{if } x = a, \\ f(x), & \text{otherwise.} \end{cases}$$

2. $H = \mathrm{introduceEdge}(H', u, v)$

   In this case, $\mathrm{dp}_H(f) = \min(\{\mathrm{dp}_{H'}(f') \;:\; f'(u) \neq f'(v) \;\wedge\; (f'(w) = f'(u) \;\vee\; f'(w) = f'(v)) \;\Rightarrow\; f(w) = f(u) = f(v)\})$.

3. $H = \mathrm{forgetVertex}(H', u)$

   Here, $\mathrm{dp}_H(f) = \min(\{\mathrm{dp}_{H'}(f') + [k = 0] \;: k \in \{0, \ldots, 4k+9\} \;\wedge\; f' = f[u \mapsto k]\})$, where $[P]$ denotes the Iverson bracket, i.e., $[P] = 1$ if P is true, and $[P] = 0$ otherwise.

4. $H = \mathrm{merge}(H', H'')$

   In this case $\partial H' = \partial H''$, and we calculate dp_H as $\mathrm{dp}_H(f) = \mathrm{dp}_{H'}(f) + dp_{H''}(f)$.

The answer is $\mathrm{dp}_T(\text{empty function}) \leqslant k$, where T is the entire decomposition.

The total complexity is bounded by

$$27^{k+1} \cdot (k+1)^{\mathcal{O}(1)} \cdot n^2 + n^{\mathcal{O}(1)} \cdot \left((4k+10)^{4k+10}\right)^2,$$

thus this algorithm is FPT when parameterized by k.